Distributed Scheduling with Apache Mesos in the Cloud

PhillyETE - April, 2015

Diptanu Gon Choudhury
@diptanu

Who am I?

- Distributed Systems/Infrastructure Engineer in the Platform Engineering Group
  - Design and develop resilient highly available services
  - IPC, Service Discovery, Application Lifecycle
- Senior Consultant at ThoughtWorks Europe
- OpenMRS/RapidSMS/ICT4D contributor

A word about Netflix

Just the stats

- 16 years
- < 2000 employees
- 50+ million users
- 5 * 10^9 hours/quarter
- Freedom and Responsibility Culture

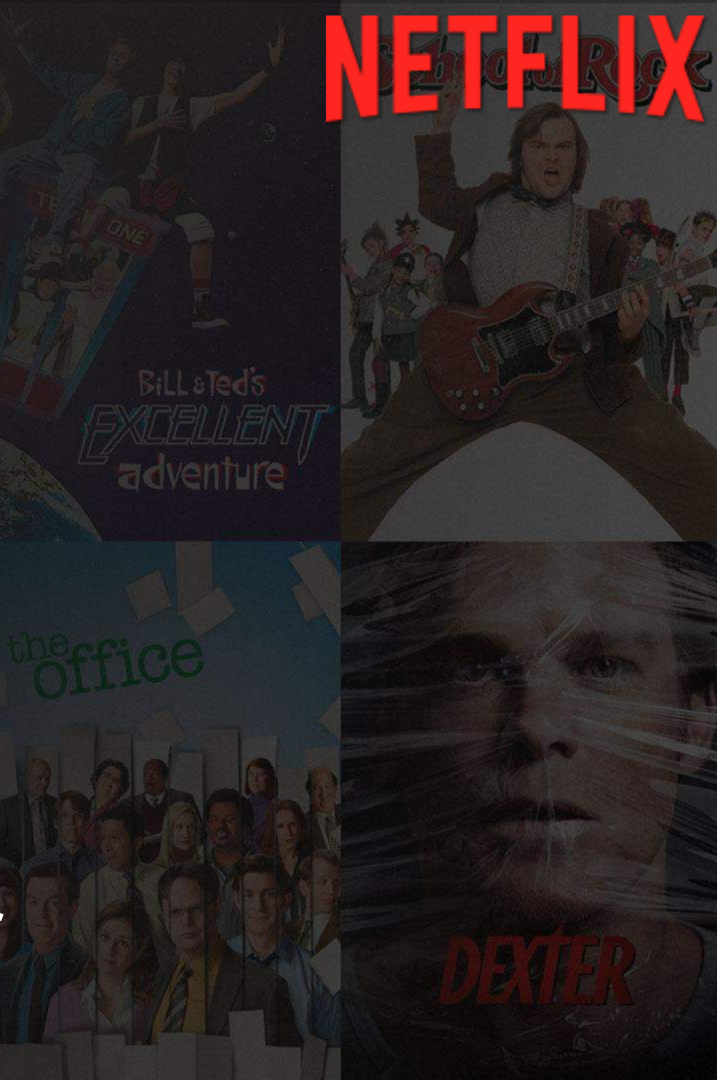
The Titan Framework

A globally distributed resource scheduler which offers compute resources as a service

Guiding Principles

Design for

- Native to the public clouds
- Availability
- Reliability
- Responsiveness
- Continuous Delivery
- Pushing to production faster



Guiding Principles

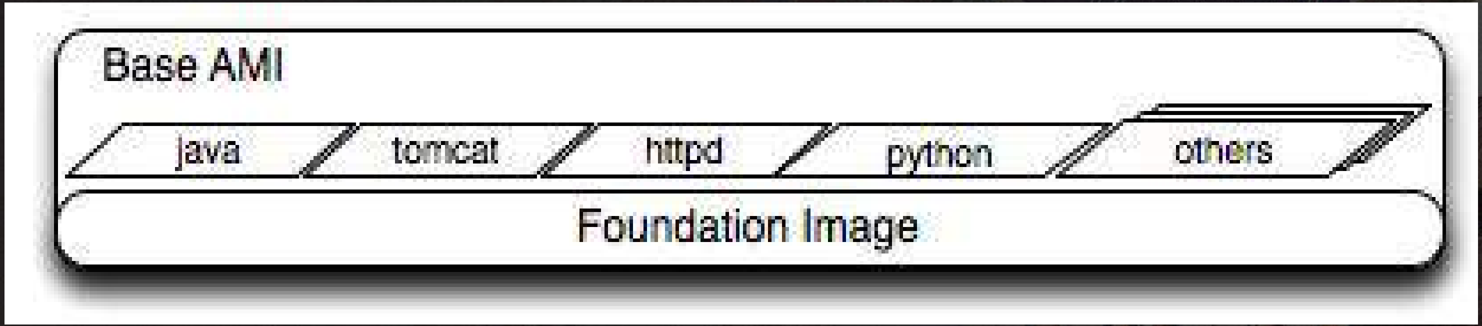
- Being able to sleep at night even when there are partial failures.
- Availability over Consistency at a higher level
- Ability for teams to fit in their domain specific needs

Active-Active Architecture

Current Deployment Pipeline

git → Bakery → ASGARD → AWS

The Base AMI



Need for a Distributed Scheduler

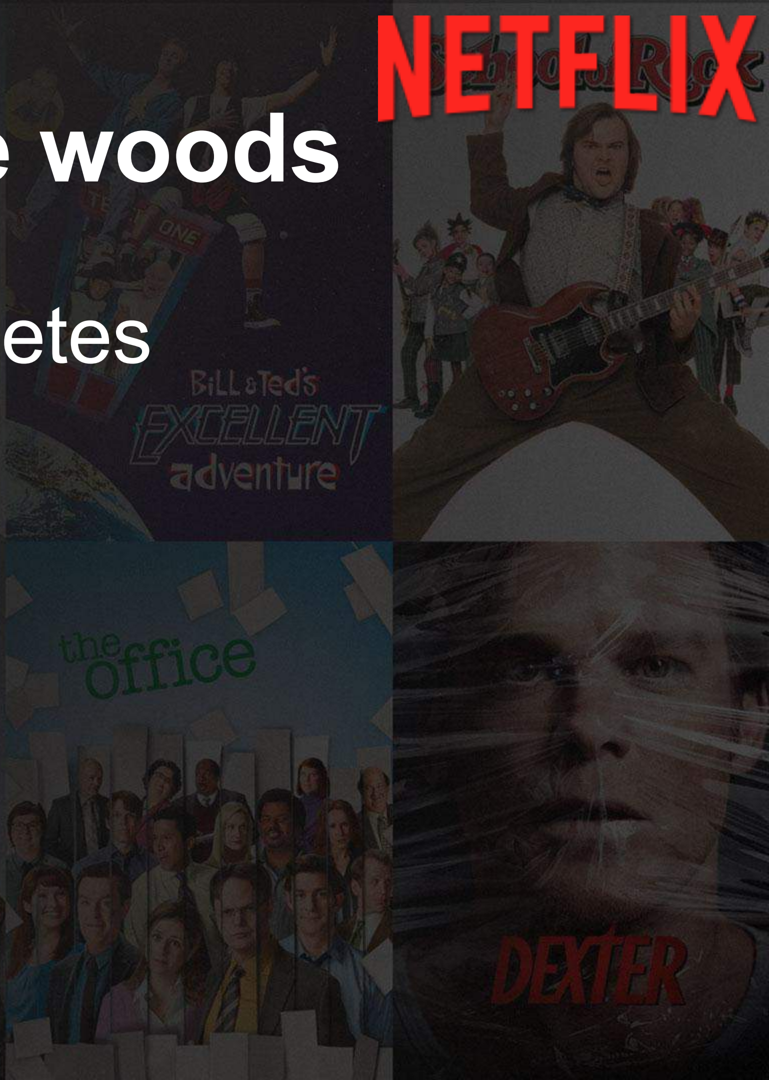
- ASGs are great for web services but for processes whose life cycle are controlled via events we needed something more flexible
- Cluster Management across multiple geographies
- Faster turnaround from development to production

Need for a Distributed Scheduler

- A runtime for polyglot development
- Tighter Integration with services like Atlas, Scryer etc

We are not alone in the woods

- Google’s Borg and Kubernetes
- Twitter’s Aurora
- Soundcloud’s Harpoon
- Facebook’s tupperware
- Mesosphere’s Marathon



Why did we write Titan

- We wanted a cloud native distributed scheduler
- Multi Geography from the get-go
- A meta scheduler which can support domain specific scheduling needs
    - Work Flow systems for batch processing workloads
    - Event driven systems
    - Resource Allocators for Samza, Spark, etc

Why did we write Titan

- Persistent Volumes and Volume Management
- Scaling rules based on metrics published by the kernel
- Levers for SREs to do region failovers and shape traffic globally

Compute Resources as a service

```
{
 “name”: “rocker”,
  “applicationName”: “nf-rocker”,
  “version”: “1.06”,
  “location”: “dc1:20,us-west-2:dc2:40,dc5:60”,
  “cpus”: 4,
  “memory”: 3200,
 “disk”: 40,
 “ports”: 2,
 “restartOnFailure”: true,
“numRetries”: 10,
“restartOnSuccess”: false
}
```

Things Titan doesn’t solve

- Service Discovery
- Distributed Tracing
- Naming Service

Building blocks

- A resource allocator
- Packaging and isolation of processes
- Scheduler
- Distribution of artifacts
- Replication across multiple geographies
- AutoScalers

Resource Allocator

- Scale to 10s of thousands of servers in a single fault domain
- Does one thing really well
- Ability to define custom resources
- Ability to write flexible schedulers
- Battle tested

Mesos

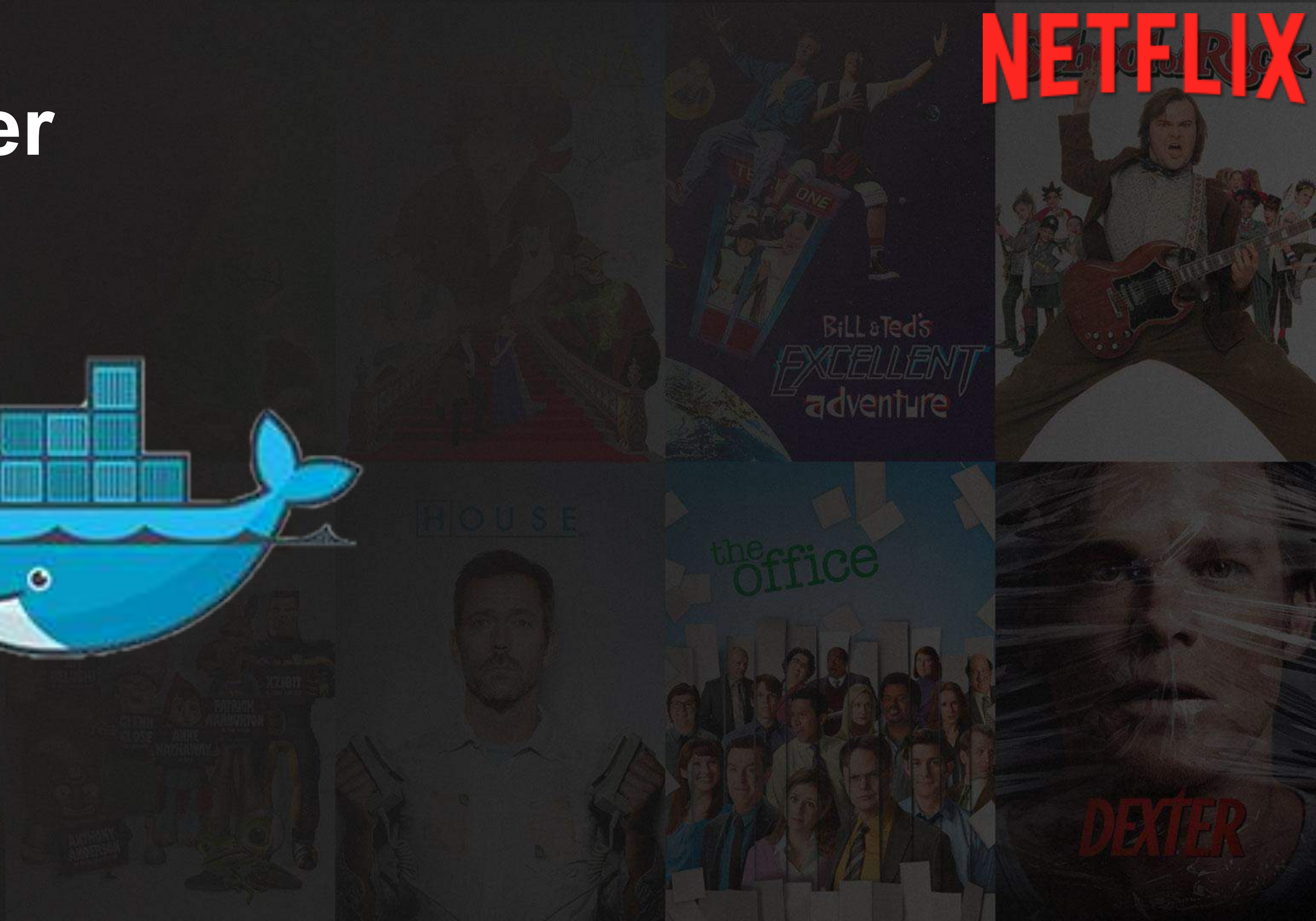
How we use Mesos

- Provides discovery of resources
- We have written a scheduler called Fenzo
- An API to launch tasks
- Allows writing executors to control the lifecycle of a task
- A mechanism to send messages

Packaging and Isolation

- We love Immutable Infrastructure
- Artifacts of applications after every build contains the runtime
- Flexible process isolation using cgroups and namespaces
- Good tooling and distribution mechanism

Docker



Building Containers

- Lots of tutorials around docker helped our engineers to pick the technology very easily
- Developers and build infrastructure uses the Docker cli to create containers.
- The docker-java plugin allows developers to think about their application as a standalone process

Volume Management

- ZFS on linux for creating volumes
- Allows us to clone, snapshot and move around volumes
- The zfs toolset is very rich
- Hoping for a better libzfs

Networking

- In AWS EC2 classic containers use the global network namespace
- Ports are allocated to containers via Mesos
- In AWS VPC, we can allocate an IP address per container via ENIs

Logging

- Logging agent on every host to allows users to stream logs
- Archive logs to S3
- Every container gets a volume for logging

Monitoring

- We push metrics published by the kernel to Atlas
- The scheduler gets a stream of metrics from every container to make scheduling decisions
- Use the cgroup notification API to alert users when a task is killed

Scheduler

- We have a pluggable scheduler called Fenzo
- Solves the problem of matching resources with tasks that are queued.

Scheduler

- Remembers the cluster state
  - Efficient bin-packing
  - Helps with Auto Scaling
  - Allows us to do things like reserve instances for specific type of workloads

Auto Scaling

- A must need for running on the cloud
- Two levels of scaling
  - Scaling of underlying resources to match the demands of processes
  - Scaling the applications based on metrics to match SLAs

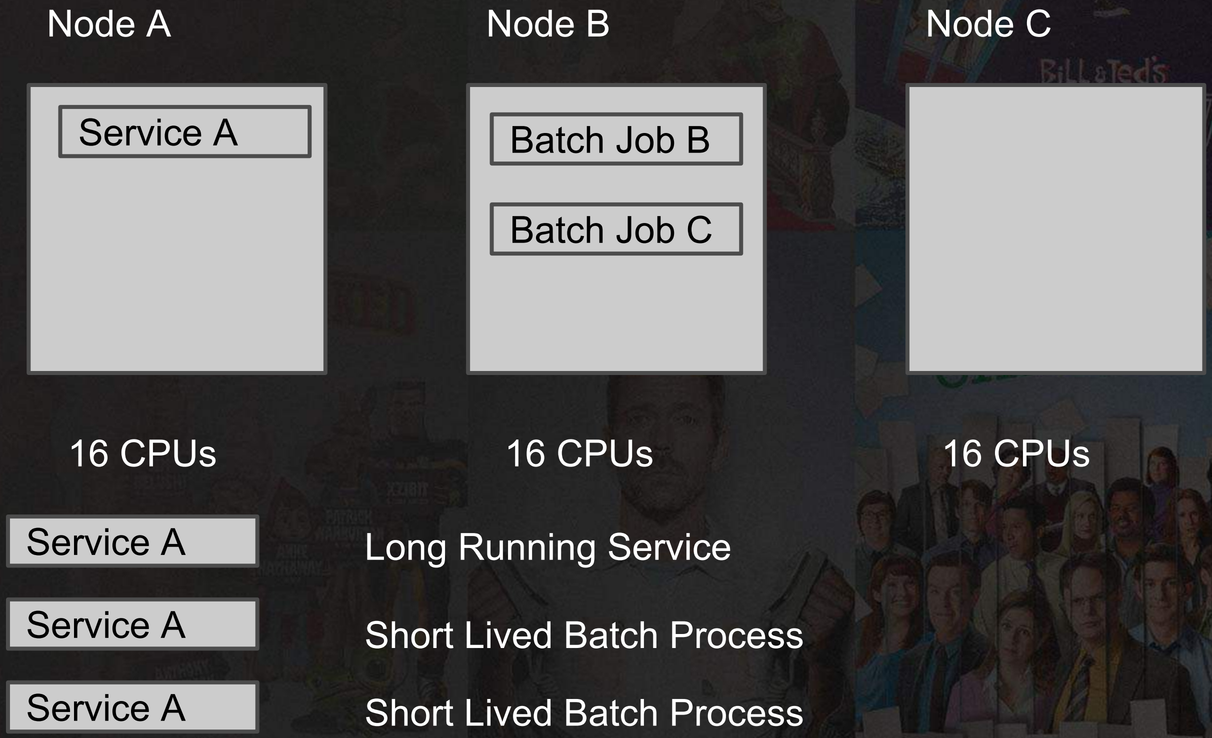
Reactive Auto Scaling

- Titan adjusts the size of the fleet to have enough compute resources to run all the tasks
- Autoscaling Providers are pluggable

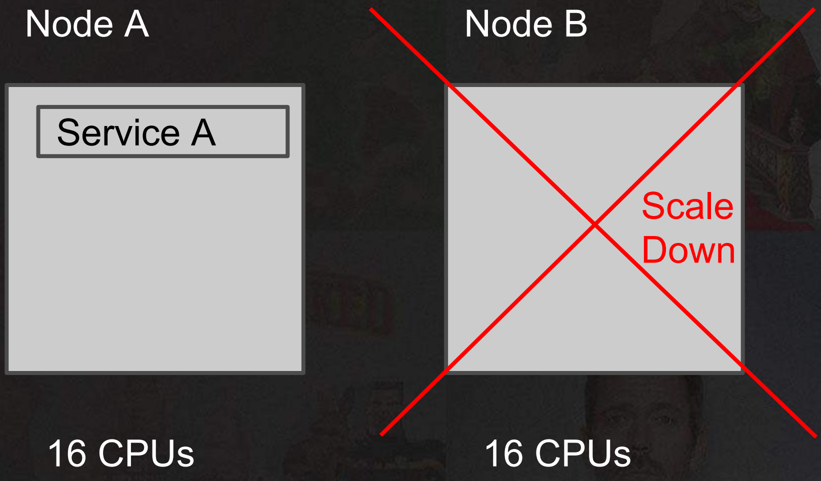
Predictive Autoscaling

- Historical data to predict the size of clusters of individual applications
- Linear Regression models for predicting near real time cluster sizes

Bin Packing for efficient Autoscaling



Bin Packing for efficient Autoscaling



Mesos Framework

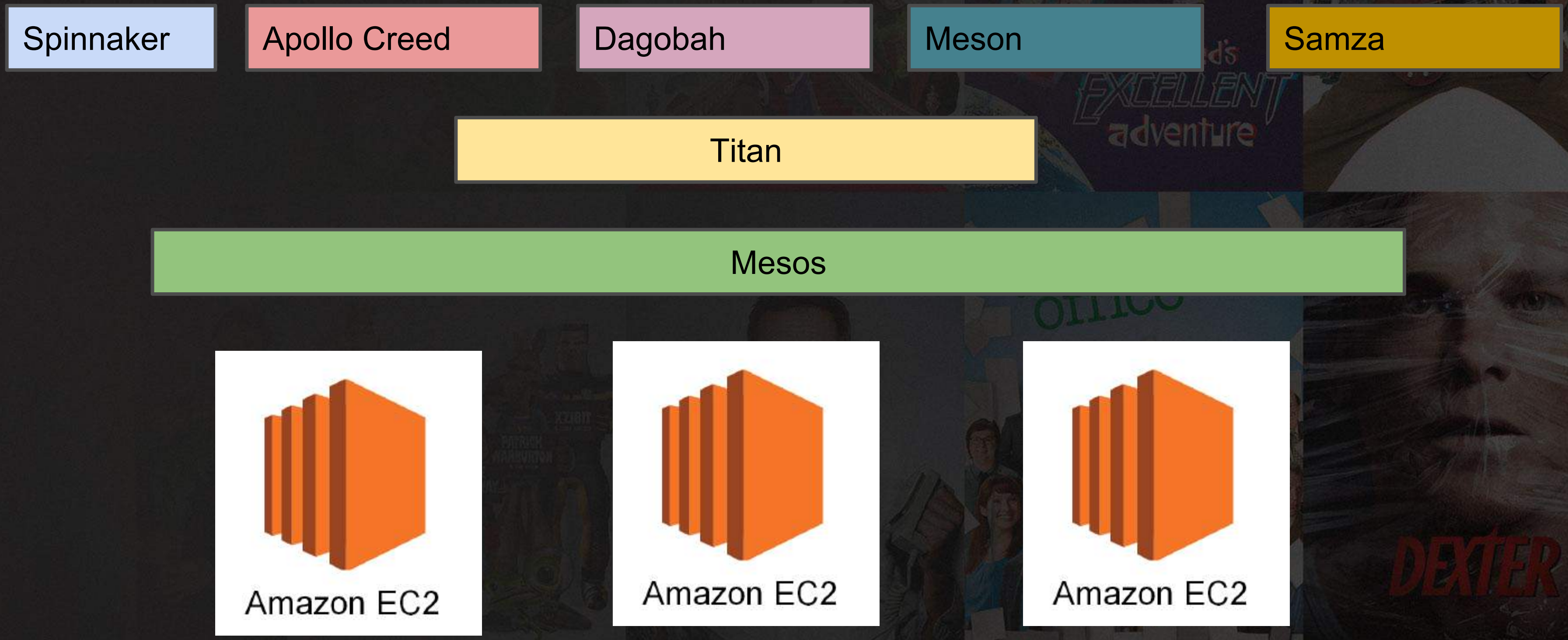
- Master Slave model with leader election for redundancy
- A single Mesos Framework per fault domain
- We currently use Zookeeper but moving to Raft
- Resilient to failures of underlying data store

Globally Distributed

- Each geography has multiple fault domains
- Single scheduler and API in each fault domain.

Globally Distributed

- All job specifications are replicated across all fault domains across all geographies
- Heart beats across all fault domains to detect failures
- Centralized control plane


Spinnaker
Apollo Creed
Dagobah
Meson
Samza
Titan
Mesos
Amazon EC2
Amazon EC2
Amazon EC2

Future

- More robust scheduling decisions
- Optimize the host OS for running containers
- More monitoring

Questions?